Claim Listing

1. (Currently amended) A system for dynamically linking application code created by a programmer into a running operating system kernel, comprising:

an environment library comprising one or more routines for insulating the application code from the operating system environment and for implementing a uniform execution environment; and

a build system for constructing a loadable module from the application code and the environment library and for constructing a standard executable program from the loadable module and an execution library ~~.~~, wherein

~~an~~ the execution library ~~comprising~~ comprises one or more routines for ~~encapsulating the loadable module within a standard executable program,~~transparently loading the loadable module into the running operating system kernel, passing arguments to the loadable module, and terminating and unloading the loadable module after receiving a termination signal~~; and~~.

~~a build system for constructing the standard executable program from the loadable module and the execution library~~

2. (Original) The system of claim 1, further comprising an infrastructure library comprising one or more routines executed prior to loading the loadable module into the running operating system kernel and/or after unloading the loadable module from the kernel.

3. (Original) The system of claim 1, wherein the execution library includes one or more routines for setting up input/output channels.

4. (Currently amended) The system of claim 1, wherein the standard executable program may be in several files or a single file.

5. (Original) A method, comprising:

creating a loadable module;

creating an executable program; and

executing the executable program, wherein the executable program performs a method comprising the steps of:

setting up input/output channels;

inserting the loadable module into an operating system address space, wherein, once the loadable the module is inserted into the operating system address space, the loadable module begins to execute; and

waiting for the loadable module to connect via kernel/user channels and then connecting those kernel/user channels to the input/output channels.

6. (Currently Amended) The method of claim 5, wherein after the loadable module is inserted into the operating system address space the loadable module performs a method comprising the steps of:

creating kernel/user channels;

creating a thread to execute ~~the~~ application code; and

waiting for the thread to complete.

7. (Original) The method of claim 6, wherein the method performed by the loadable module further includes the step of freeing resources after the thread completes.

8-9. Cancelled

10. (New) The system of claim 1, wherein one of the one or more routines of the execution library includes code for executing a utility for installing the loadable module into the running operating system kernel.

11. (New) The system of claim 10, wherein the utility for installing the loadable module into the running operating system kernel is the insmod program.

12. (New) The system of claim 1, wherein the build system includes instructions for compiling the application code into object code.

13. (New) The system of claim 12, wherein the build system further includes instructions for linking said object code with object code from the environment library to produce a linked object module.

14. (New) The system of claim 13, wherein the build system further includes instructions for converting the linked object module into a C code array.

15. (New) The system of claim 13, wherein the build system further includes instructions for compiling the C code array to produce an object file and for linking said object file with object code from the execution library to produce the standard executable program.

16. (New) The system of claim 1, wherein the environment library includes one or more routines to create kernel/user channels.

17. (New) The system of claim 1, wherein the environment

library includes one or more routines to create a thread to execute the application code.

18. (New) The system of claim 17, wherein the environment library includes one or more routines for freeing resources and unloading the loadable module when the thread completes.

19. (New) The system of claim 1, wherein the environment library includes one or more routines for (a) copying in arguments; (b) creating communication channels that connect the loadable module to the executable program; (c) requesting a block of memory from the operating system and storing a structure therein that describes the application code; and (d) putting data describing the application code on a task list.

20. (New) The system of claim 19, wherein the environment library further includes one or more routines for (a) removing said data describing the application code from the task list; (b) closing said communication channels that connect the loadable module to the executable program; and (c) freeing the block of memory that was requested from the operating system.

21. (New) A computer readable medium having computer instructions stored thereon, the computer instructions comprising:

a first set of computer instructions for insulating application code from an operating system environment;

a second set of computer instructions for constructing a loadable module from the application code and the first set of computer instructions; and

a third set of computer instructions for constructing an

executable program from the loadable module and a fourth set of computer instructions; wherein

the fourth set of computer instructions includes computer instructions for transparently loading the loadable module into a running operating system kernel, passing arguments to the loadable module, and terminating and unloading the loadable module from the running operating system kernel after receiving a termination signal.

22. (New) The computer readable medium of claim 21, wherein the computer instructions for loading the loadable module into the running operating system kernel include computer instructions for executing a utility for installing the loadable module into the running operating system kernel.

23. (New) The computer readable medium of claim 22, wherein the utility for installing the loadable module into the running operating system kernel is the insmod program.

24. (New) The computer readable medium of claim 21, wherein the second set of computer instructions includes instructions for compiling the application code into object code.

25. (New) The computer readable medium of claim 24, wherein the second set of computer instructions further includes instructions for linking said object code with object code from the environment library to produce a linked object module.

26. (New) The computer readable medium of claim 25, wherein the third set of computer instructions includes

instructions for converting the linked object module into a C code array.

27. (New) The computer readable medium of claim 26, wherein the third set computer instructions of further includes instructions for compiling the C code array to produce an object file and for linking said object file with object code from a library to produce the executable program.

28. (New) The computer readable medium of claim 21, wherein the first set of computer instructions includes instructions for creating kernel/user channels.

29. (New) The computer readable medium of claim 21, wherein the first set of computer instructions includes instructions for creating a thread to execute the application code.

30. (New) The computer readable medium of claim 29, wherein the first set of computer instructions includes instructions for freeing resources and unloading the loadable module when the thread completes.

31. (New) The computer readable medium of claim 21, wherein the first set of computer instructions includes instructions for (a) creating communication channels that connect the loadable module to the executable program; (b) requesting a block of memory from the operating system and storing a structure therein that describes the application code; and (c) putting data describing the application code on a task list.

32. (New) The computer readable medium of claim 31,

wherein the first set of computer instructions further includes instructions for (a) removing said data describing the application code from the task list; (b) closing said communication channels that connect the loadable module to the executable program; and (c) freeing the block of memory that was requested from the operating system.

33. (New) The computer readable medium of claim 21, wherein the executable program may be in several files or a single file.

34. (New) A computer system, comprising:

first means for insulating application code from an operating system environment;

second means for constructing a loadable module from the application code and the first means;

third means for constructing an executable program from the loadable module; and

fourth means for transparently loading the loadable module into a running operating system kernel, passing arguments to the loadable module, and terminating and unloading the loadable module from the running operating system kernel after receiving a termination signal.

35. (New) The computer system of claim 34, wherein means for loading the loadable module into the running operating system kernel include means for executing a utility for installing the loadable module into the running operating system kernel.

36. (New) The computer system of claim 35, wherein the utility for installing the loadable module into the running

operating system kernel is the insmod program.

37. (New) The computer system of claim 34, wherein the second means includes means for compiling the application code into object code.

38. (New) The computer system of claim 37, wherein the second means further includes means for linking said object code with object code from the environment library to produce a linked object module.

39. (New) The computer system of claim 38, wherein the third means includes means for converting the linked object module into a C code array.

40. (New) The computer system of claim 39, wherein the third means further includes instructions for compiling the C code array to produce an object file and for linking said object file with object code from a library to produce the executable program.

41. (New) The computer system of claim 34, wherein the first means includes means for creating kernel/user channels.

42. (New) The computer system of claim 34, wherein the first means includes means for creating a thread to execute the application code.

43. (New) The computer system of claim 42, wherein the first means includes means for freeing resources and unloading the loadable module when the thread completes.

44. (New) The computer system of claim 34, wherein the

first means includes means for (a) creating communication channels that connect the loadable module to the executable program; (b) requesting a block of memory from the operating system and storing a structure therein that describes the application code; and (c) putting data describing the application code on a task list.

45. (New) The computer system of claim 44, wherein the first means further includes means for (a) removing said data describing the application code from the task list; (b) closing said communication channels that connect the loadable module to the executable program; and (c) freeing the block of memory that was requested from the operating system.

46. (New) The computer system of claim 34, wherein the executable program may be in several files or a single file.

47. (New) A computer system for dynamically linking application code created by a programmer into a running operating system kernel, comprising:

means for creating a loadable module; and

means for creating an executable program that is configured to performs a method comprising the steps of:

setting up input/output channels;

inserting the loadable module into address space of the running operating system kernel, wherein, once the loadable the module is inserted into the address space, the loadable module begins to execute; and

waiting for the loadable module to connect via kernel/user channels and then connecting those kernel/user channels to the input/output channels.

48. (New) The computer system of claim 47, wherein the

loadable module is configured to perform a method after the loadable module is inserted into the operating system address space, wherein said method comprises the steps of:

creating kernel/user channels;

creating a thread to execute the application code; and

waiting for the thread to complete.

49. (New) The computer system of claim 48, wherein the method performed by the loadable module further includes the step of freeing resources after the thread completes.

50. (New) The computer system of claim 47, wherein the step of inserting the loadable module into an operating system address space includes the step of creating a child process, wherein the child process replaces its image with the insmod process image.

51. (New) The computer system of claim 50, wherein the step of inserting the loadable module into an operating system address space further includes the step of piping the loadable module to the insmod process.

52. (New) A method for dynamically linking application code created by a user into a running operating system kernel, comprising:

constructing a loadable module from application source code written by a user;

creating an executable program, wherein the executable program is configured to transparently load the loadable module into the running operating system kernel;

executing the executable program, thereby loading the loadable module into the running operating system kernel; and

unloading the loadable module from the running operating

system kernel by sending a termination signal to the executable program.

53. (New) The method of claim 52, wherein the application source code is an ordinary application program.

54. (New) The method of claim 52, wherein the step of constructing the loadable module from the application source code consists essentially of executing a pre-defined makefile.

55. (New) The method of claim 52, further comprising the step of providing a makefile to the user, wherein the user performs the step of constructing the loadable module by executing the makefile after the user has created the application code.

56. (New) The method of claim 52, further comprising the step of providing the user with a library comprising object code, wherein the step of constructing the loadable module from the application source code comprises the steps of compiling the application source code into object code; linking the object code with object code from the library to produce a linked object module; and converting the linked object module into a C code array.

57. (New) The method of claim 56, wherein the step of constructing the loadable module further comprises the step of compiling the C code array to produce an object file.

58. (New) The method of claim 57, further comprising the step of providing the user with a second library comprising object code, wherein the step of constructing the executable program comprises the steps of linking the object file with

object code from the second library.

59. (New) The method of claim 56, wherein the library includes one or more routines to create kernel/user channels.

60. (New) The method of claim 56, wherein the library includes one or more routines to create a thread to execute the application code.

61. (New) The method of claim 60, wherein the library includes one or more routines for freeing resources and unloading the loadable module when the thread completes.

62. (New) The method of claim 56, wherein the library includes one or more routines for (a) copying in arguments; (b) creating communication channels that connect the loadable module to the executable program; (c) requesting a block of memory from the operating system and storing a structure therein that describes the application code; and (d) putting data describing the application code on a task list.

63. (New) The method of claim 62, wherein the environment library further includes one or more routines for (a) removing said data describing the application code from the task list; (b) closing said communication channels that connect the loadable module to the executable program; and (c) freeing the block of memory that was requested from the operating system.

64. (New) The method of claim 52, wherein the executable program is configured to set up input/output channels.

65. (New) The method of claim 52, wherein the executable

program is configured to execute a utility for installing the loadable module into the running operating system kernel.

66. (New) The method of claim 65, wherein the utility for installing the loadable module into the running operating system kernel is the insmod program.

67. (New) The method of claim 65, wherein the step of inserting the loadable module into an operating system address space includes the step of creating a child process, wherein the child process replaces its image with the insmod process image.

68. (New) The method of claim 67, wherein the step of inserting the loadable module into an operating system address space further includes the step of piping the loadable module to the insmod process.